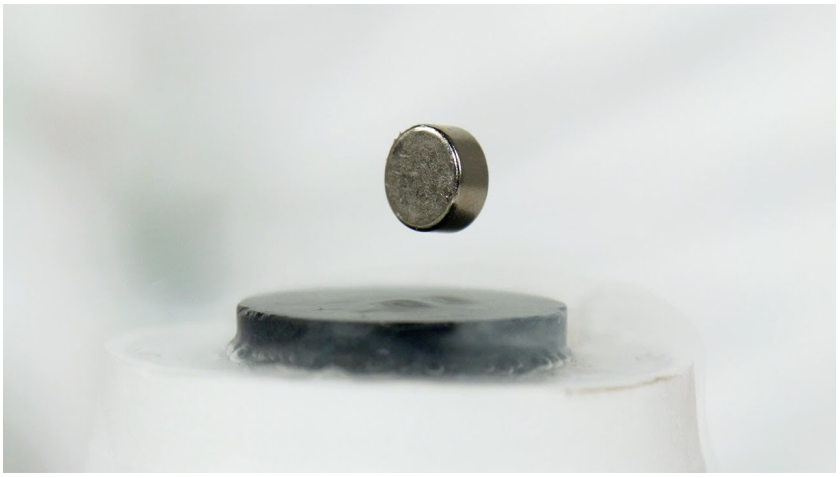
Ciências ULisboa
Faculdade de Ciências da Universidade de Lisboa

Prediction of Critical Temperature of Superconductors

Universidade de Lisboa
Data Science



Student:
Timothée Belime N.º55959

Teachers:
André Moitinho de Almeida

Contents

1 Introduction

In this project we attempt to find a model which would allow us to better predict the critical temperature of superconductors. Superconductors are of real interest for the industry as they present practaclly zero resistance to current flow. We will not attempt to give any theoretical explanation over how nature operates, but will rather try to extract as much information as we can from collected data of 21.263 superconductors, with 81 physical features each and their respective critical temperature, T_c. These features include the $number\_of\_elements$ + 10 statistical formulas applied to 8 different properties of each material.

Variable	Units	Description
Atomic Mass	atomic mass units (AMU)	total proton and neutron rest masses
First Ionization Energy	kilo-Joules per mole (kJ/mol)	energy required to remove a valence electron
Atomic Radius	picometer (pm)	calculated atomic radius
Density	kilograms per meters cubed (kg/m^3)	density at standard temperature and pressure
Electron Affinity	kilo-Joules per mole (kJ/mol)	energy required to add an electron to a neutral atom
Fusion Heat	kilo-Joules per mole (kJ/mol)	energy to change from solid to liquid without temperature change
Thermal Conductivity	watts per meter-Kelvin (W/(m $\times$ K))	thermal conductivity coefficient κ
Valence	no units	typical number of chemical bonds formed by the element

Feature & Description	Formula
Mean	$= \mu = (t_1 + t_2)/2$
Weighted mean	$= \nu = (p_1 t_1) + (p_2 t_2)$
Geometric mean	$= (t_1 t_2)^{1/2}$
Weighted geometric mean	$= (t_1)^{p_1} (t_2)^{p_2}$
Entropy	$= -w_1 \ln(w_1) - w_2 \ln(w_2)$
Weighted entropy	$= -A \ln(A) - B \ln(B)$
Range	$= t_1 - t_2 \ (t_1 > t_2)$
Weighted range	$= p_1 t_1 - p_2 t_2$
Standard deviation	$= [(1/2)((t_1 - \mu)^2 + (t_2 - \mu)^2)]^{1/2}$
Weighted standard deviation	$= [p_1 (t_1 - \nu)^2 + p_2 (t_2 - \nu)^2)]^{1/2}$

Figure 1: 8 physical measurements of materials, times 10 statistical properties, giving 80 features for each material.

The data was downloaded from *UC Irvine Machine Learning Repository* [1] in a .csv file. We used the XGBoost regression technique inspired on the paper of *Kam Hamidieh* [2] and implemented it with the help of *XGBoost Python Package*[3] and *Scikit learn*, as well as article [4].

We will start by doing a Principal Component Analysis, to see the dimension actually spanned by the data, and see if some visualization of the data along the main principle

PC1	0.389	PC6	0.038	PC11	0.018
PC2	0.105	PC7	0.036	PC12	0.015
PC3	0.095	PC8	0.031	PC13	0.012
PC4	0.079	PC9	0.024	PC14	0.010
PC5	0.059	PC10	0.020	PC15	0.010

Table 1: 15 main principle components, enough to explain 94.03% of the data.

components can give us some preliminary information about the behavior of T_c. We than perform XGBoost regression and k-fold cross validation in order to find our model and see how well it performs.

All the coding was made on Python and can be consulted at the following GitHub repository [5].

2 Data Analysis

2.1 Dimensionality reduction and classification

As we are in a 81 dimensional dataset each with properties of different units and scales, we started by performing a normalization, as to get mean value of 0 and standard deviation of 1 for each feature. Using Scikitlearn, we performed Principal Component Analysis to see if we could reduce dimensionality and better understand the structure of the data.

2.1.1 Principal Component Analysis

We found that 99% of the variance can be explained with 31 components instead of 81, and 94% with only 15 components. (fig. 2)

It would be quite reasonable to think of reducing the dimensionality of the data to 15 but for the small amount of calculations we will be performing we can keep 31 components without thinking of computational cost.

In order to better understand the data, we observe if any of the original features dictates predominantly compared to others by analysing the coordinates of PC1 and PC2. Furthermore we will plot a 3D and 2D projection of the data with the new components, in order to see if clustering could be of any help in classifying the data.

PC1 has cordinate -0.16 of $range\_fie$ (range of first ionization energy) component, which is the most we have of any of the 81 components. Because PC1 is normalized, we have that $range\_fie$ accounts for $\frac{-0.16^2}{1^2} = 0.0256$ of PC1, which in turn accounts for 0.39 of the variance, meaning that $range\_fie$ explains $(0.0256 * 0.39 = 0.01)$ 1% of the variance along the PC1 axes. Adding the fact that it is not much more relevant than other features

which have the same relevance in PC1, we conclude that none of the features really dictate the behavior of the data.

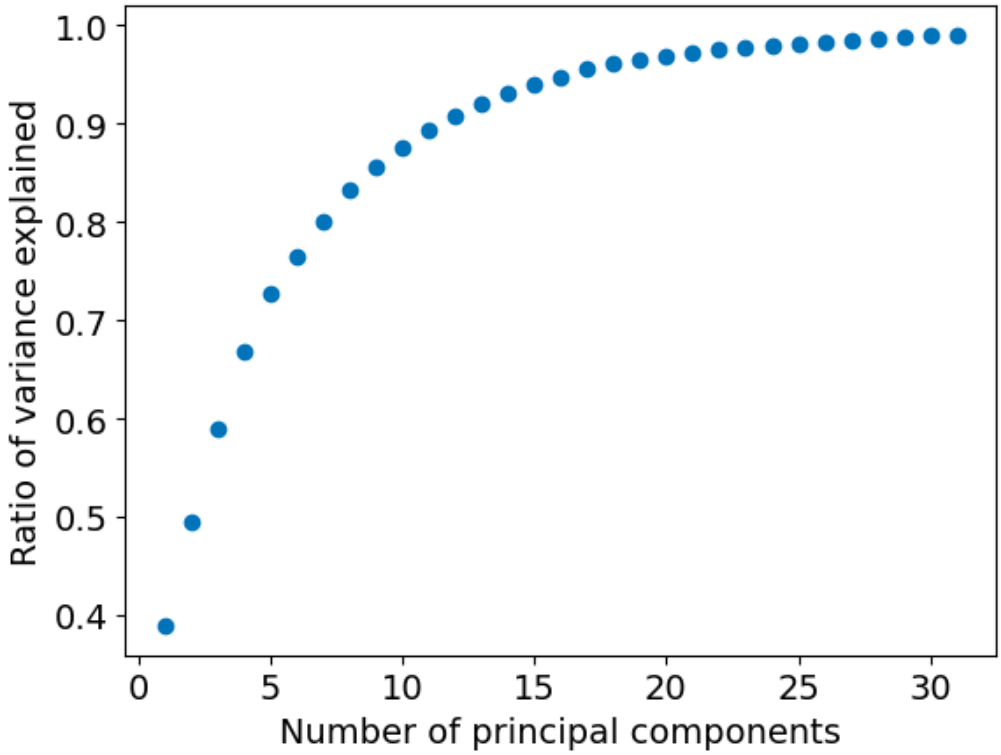


Figure 2: With only the first 2 components 49% of the variance is explained. 99% of it is explained with 31 components instead of 81, and 94% with only 15 components.

We proceed by plotting 1000 data points randomly selected from the dataset in 3D and 2D using PC 1, 2 and 3 (see fig. 3). We see that the data seems to split around the origin of PC1, with high temperature superconductors on the negative side of PC1, and low temperature superconductors on the positive side of PC1. High temperature superconductors were discovered in the late 80s and are known for having slightly different behaviors than low temperature superconductors, which would explain the separation in two groups.

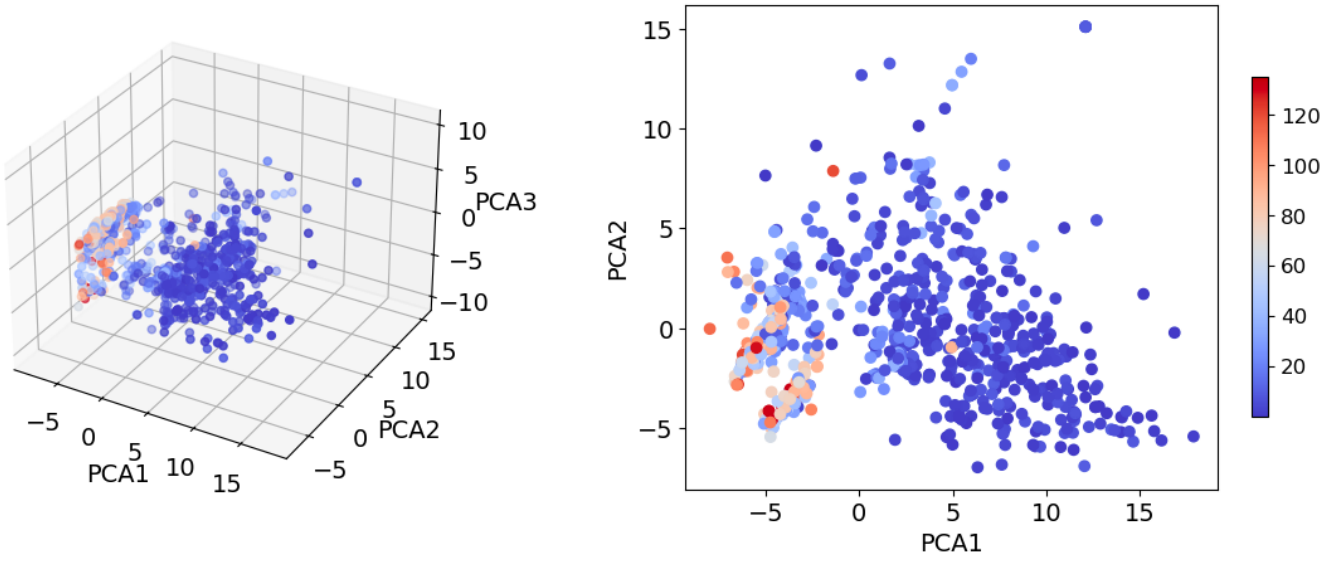


Figure 3: 1000 samples of the PCA resulting dataset, projected to the first 3, and 2, principal components.

2.1.2 Kmeans clustering

We decide to apply k-means clustering to the reduced dataset obtained thru PCA (dimension 31). Assuming there are 2 clusters, we perform kmeans and visualize the resulting clusters projecting 10000 samples, randomly picked from the dataset, on PC1 and PC2 (fig. 4). The result confirms that the superconductors can be classified in 2 different types, with a clear cut slightly over $PC1 = 0$. We then observe another division in negative side of PC1 (two diagonal bulbs), which could mean further subclassification is possible inside that class.

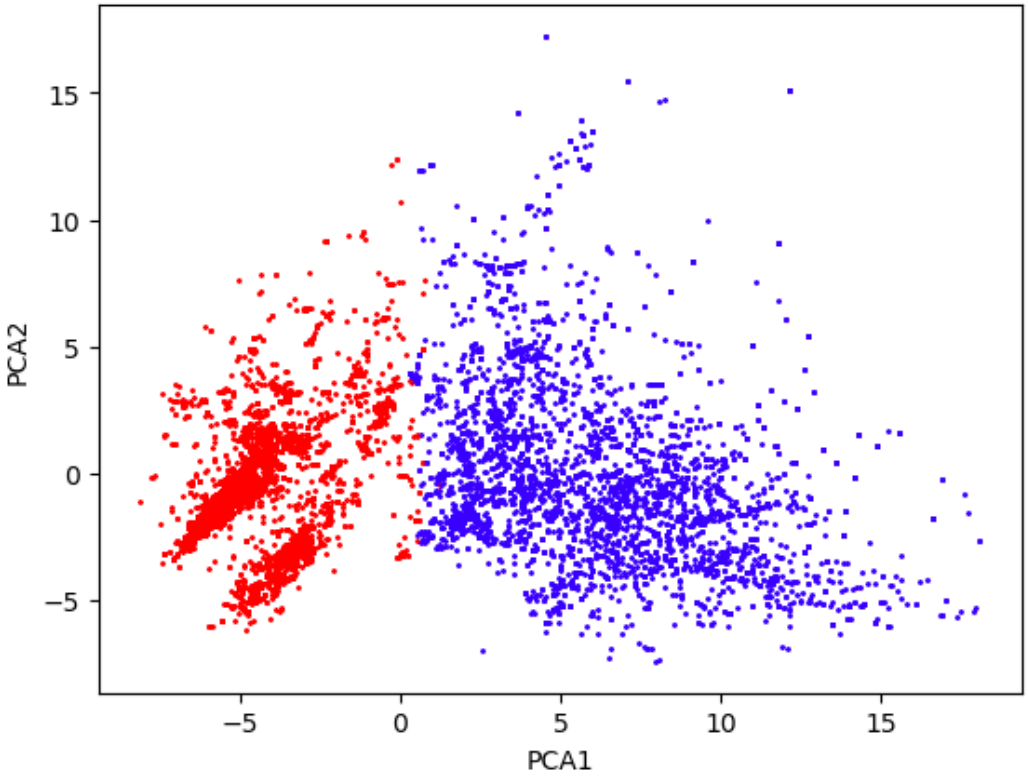


Figure 4: 10000 samples of the PCA resulting dataset, projected on PC1 and PC2, with the respective color of their cluster.

2.2 Critical temperature prediction

In order to find our prediction model based on the data we have, we start by splitting it in a training subset of 70% of the total samples, and a testing subset of 30%, selected randomly and without replacement. We will first test if the kmeans we used can serve as a good classifier, then we will use the extreme gradient method of XGBoost. This method has gained a lot of popularity for its efficiency on apparently unstructured datasets, it works with a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. It works for both regression and classification problems, and has many interesting features. In our case we will stick to the default squared error regression feature.

2.2.1 Kmeans classification

We use the training subset to fit and predict a Kmeans model to then attribute a cluster to each sample of the testing subset. We then compare if the clusters attributed by this model correspond to the ones attributed by the model we obtained by fitting the entire dataset (fig. 4).
We calculated the ratio of failure by suming all the samples that resulted in a different

cluster and dividing by the total testing samples. The result was either 1 or 0, depending on the label attributed to each cluster, meaning the success ratio is 1. So this method allows us to confidently classify the superconductors in two different categories. To better understand these categories further study should be undertaken but for the sake of this work we will simply say that there are 2 categories with a strong correlation to high and low temperature superconductors.

2.2.2 XGBoost and 10-fold Cross Validation

Using again a $70 - 30$ split of the dataset (this time the original unscaled dataset) we applied the XGBoost method with different parameter values for "colsample_bytree", which means that only a fraction of the features is selected to build each of the trees; "num_boost_round", which determines the number of trees thru which the data will be boosted; and "subsample", which selects, for each tree, only a fraction of the data to be processed. This last parameter was left to its default value 1 after finding out with 10-fold cross validation that it didn't add any progress.
The data for this method was not normalized, because it turned out unnecessary, and we tested it both ways to make sure it was. In order to evaluate the model we used root mean squared error, RMSE, and performed 10-fold cross validation built-in method to the training set. After setting "num_boost_round" to 100, we saw that the RMSE has an exponenial decay and decided to set the number of rounds to 50. We see a tendency to overfit as we increase the number of rounds. We then tried different values of "colsample_bytree" from 0.2 to 0.8 and saw that standard deviaton of training RMSE was smaller and more stable for values of 0.3, but standard deviaton of testing RMSE was slightly higher then for othe values of the parameter. We are interested in getting the smallest test mean and standard deviation RMSE, so in the end we chose to set "colsample_bytree" to 0.5.

With those parameters set, we then tested the model on the remaining 30% of the data and compared the predicted T_c to the known T_c from the dataset (fig. 5).

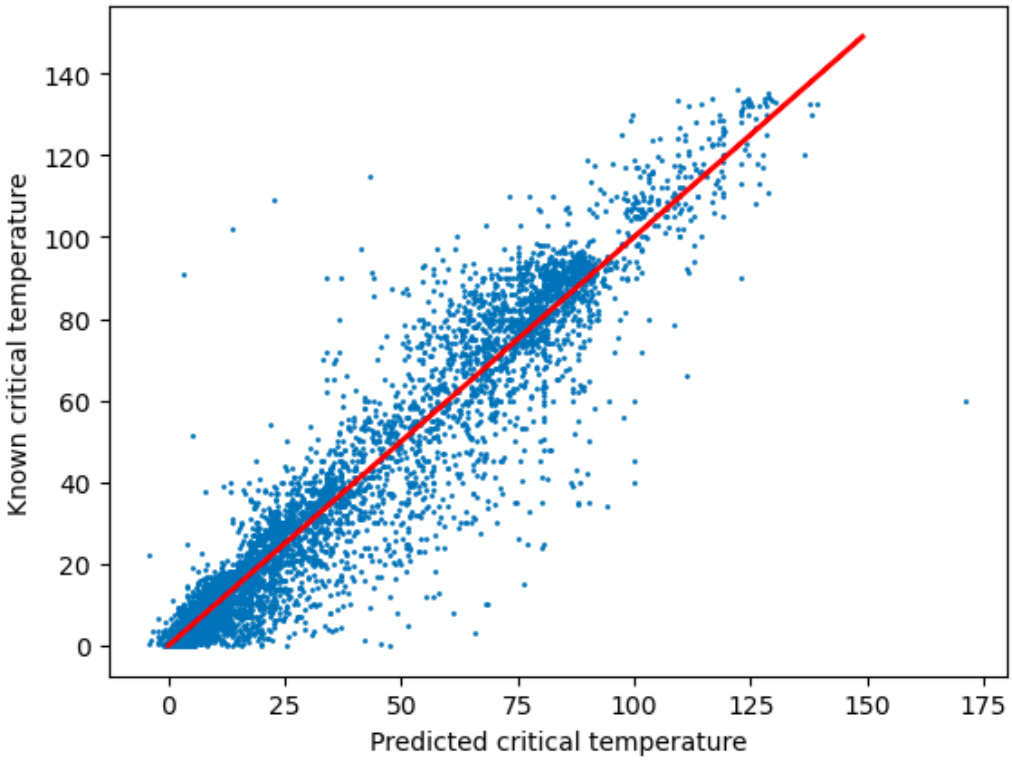


Figure 5: Test subset predictions versus known values of T_c. We see some values are pretty far out: 5 are very far from doing a good prediction but most of them are roughly within a 25% margin error.

By plotting the feature importance we find that features 1, 6, 2, 8, and 70 have better scores than the others, they respectively correspond to "*mean_atomic_mass*", "*wtd_entropy_atomic_mass*","*wtd_mean_atomic_mass*","*wtd_range_atomic_mass*", *and* "*wtd_std_ThermalConductivity*".

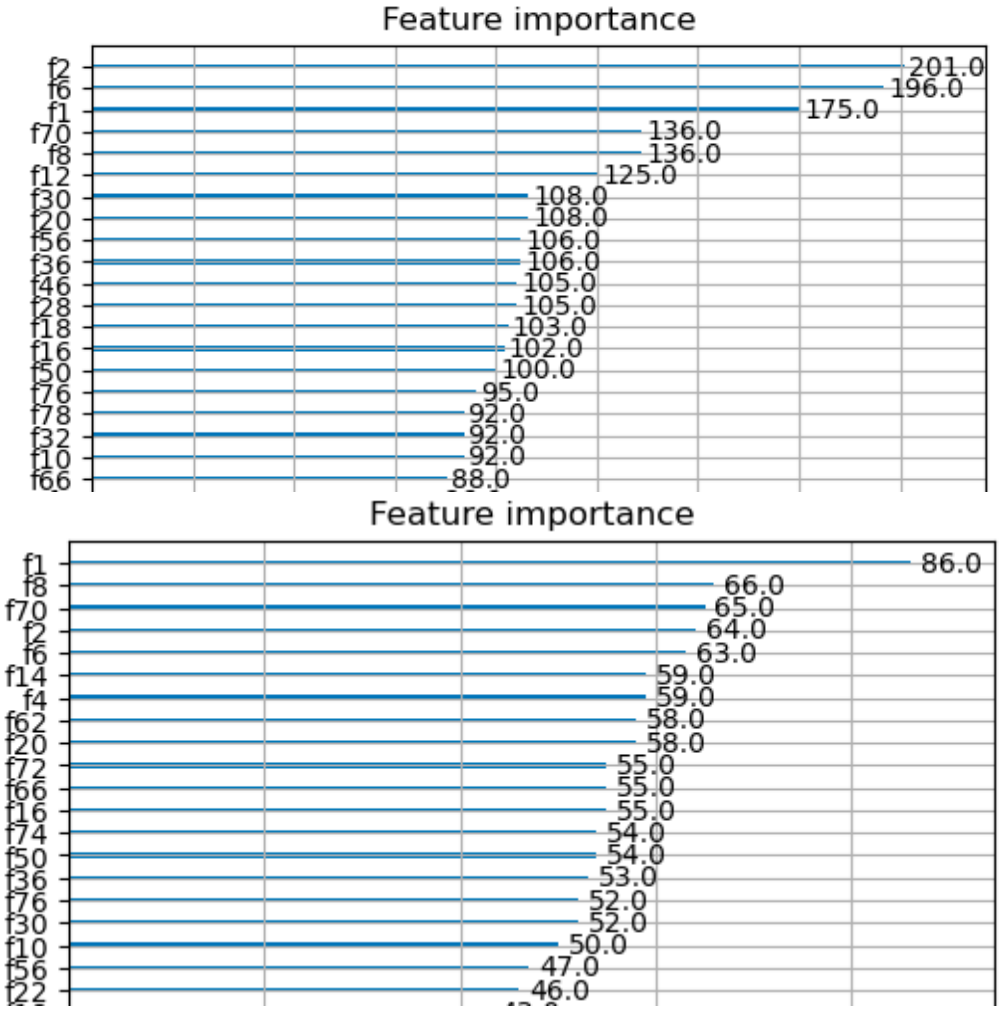


Figure 6: On top: feature importance after 100 rounds run with default "$colsample\_bytree$" = 1. At the bottom: feature importance after 50 rounds run with "$colsample\_bytree$" = 0.5.

3 Conclusion

We conclude that atomic mass is quite relevant in the prediction of T_c but that it would not be enough to make a good prediction. Just as PCA has shown us, the dimensionality can be reduced to 31 by loosing only 1% of information on the variance, but almost all features seem usefull to the model. The least relevant, according to XGBoost method, being "$range\_fie$","$gmean\_Valence$", "$range\_Valence$","$range\_Density$", and"$range\_ElectronAffinity$" (see fig. 7).
The model defined with XGBoost, with 50 estimators and "$colsample\_bytree$" = 0.5 provides us with predictions of $RMSE = 10.4 \pm 0.5$.

We also conclude that kmeans clustering after applying PCA and reducing dimensionality provides us with a good model to classify superconductors into two categories: the ones on negative PC1 and the ones on positive PC1. This classification seems to be strongly correlated with T_c but we will stay precautious and just say that further investigation would need to be done to better understand the causes and consequences of this classification.

Figure 7: Last scores of feature importance after running our model.

References

[1] **UC Machine Learning Repository**
https://archive.ics.uci.edu/dataset/464/superconductivty+data

[2] **Paper of *Kam Hamidieh***
https://github.com/khamidieh/predict_tc/blob/master/paper_3.pdf

[3] **XGBoost documentation**
https://xgboost.readthedocs.io/en/stable/

[4] **Article by *Prashant Banerjee***
https://www.kaggle.com/code/prashant111/xgboost-k-fold-cv-feature-importance#5.-k-fold-Cross-Validation-using-XGBoost-

[5] **GitHub repository containing Jupyter Notebook**
https://github.com/timbelime/data_science